Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Profs. Archana Chidanandan and J.P. Mellor

Homework 2
(Exceptions, Register Transfer Language and Datapaths)
Maximum points: 55 points + 15 extra credit points

Directions

This assignment is due Tuesday, Oct. 12th for Sections 1, 2 and 3. Submit your solutions on a separate sheet of paper.

Learning Objectives

In the process of completing this homework assignment, students will develop their abilities to

- Describe the implementation of machine language instructions using Register Transfer Language.
- Identify the components required to implement machine language instructions, including their input, output, and control signals, as well as their high-level behavior.
- Design datapaths to support machine language instruction sets by specifying the interconnections between components and the behavior of the associated control units.
- Determine how to handle exceptions, including interrupts.

Problems

1. Figure 5.28 of Patterson & Hennessy, shows a complete datapath for a multicycle implementation of most R-Type MIPS assembly language instructions, as well as `lw`, `sw`, `beq`, and `j`. For this question, you are required to modify the datapath to include the multicycle implementation of the `bne` instruction. Specifically, you must:

   a. [5 points] Write a multicycle RTL description of an implementation of the `bne` instruction that uses as few cycles as possible without extending the clock cycle of your design.

   b. [5 points] List all new and modified components required for the implementation of the `bne` instruction. Also, list the input, output, and control signals for each of those components. Indicate the number of bits in each signal.

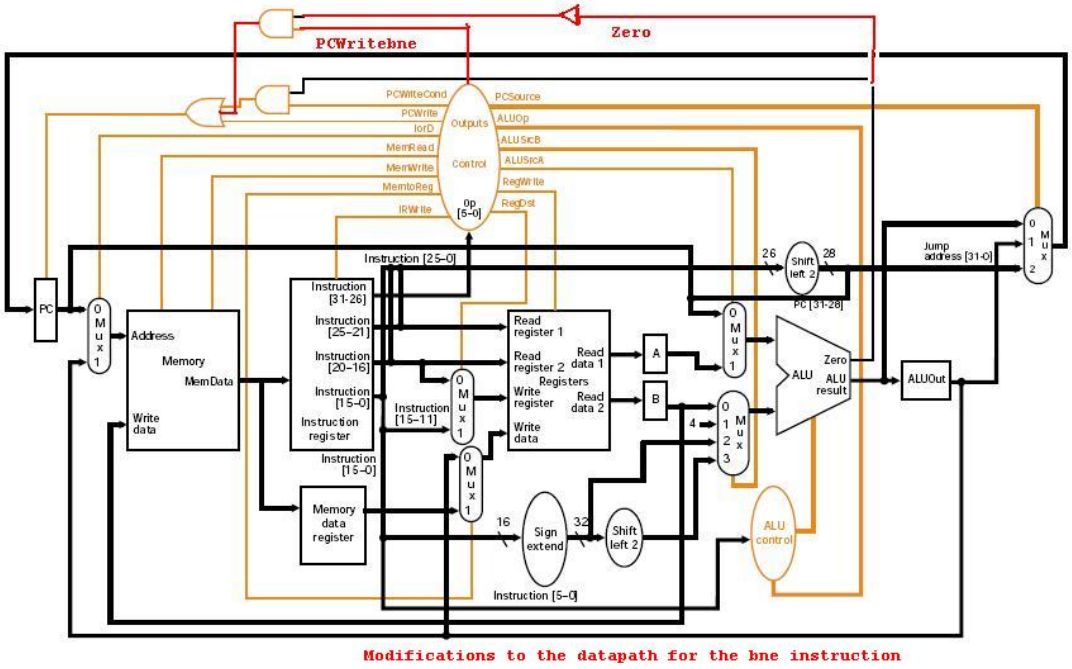
   c. [5 points] Add any necessary datapath components and control signals to the multicycle datapath. A pdf version of Figure 5.28 is available on the class website(Homework).

   a. RTL for the bne instruction

```
1. PC = PC + 4; IR = Mem[PC]
2.  A = Reg[IR[25:21]];   B = Reg[IR[20:16]];
   ALUOut = PC + (SE[IR[15:0]] << 2);
   If(IR[31:26]==5) then
3. If (A – B != 0) PC = ALUOut
```

   b. An AND gate and an inverter are required.

c. The following datapath shows the changes required to implement the MIPS `bne` instruction.



Modifications to the datapath for the bne instruction

2. [15 points] Repeat Steps 1a, 1b and 1c for the `mfc0` and `mtc0` MIPS instructions (combine the component lists and the datapath modifications). *Hint:* Page A71 (on the CD) of Hennessey and Patterson, has a description for these two instructions.

d. RTL description for `mfc0` and `mtc0`

1. `mfc0 rt rd` - Move the contents of co-processor 0's register rd to register rt.

```
1. PC = PC + 4; IR = Mem[PC]
2. A = Reg[IR[25:21]];   B = Reg[IR[20:16]];
   ALUOut = PC + (SE[IR[15:0]] << 2);
   If((IR[31:26]==16)and(IR[25:21]==0)) then
3. D = CoP0Reg[IR[15:11]]; //Could do this in clock
                           //cycle 2 and save a clock cycle
4. Reg[IR[20:16]] = D;
```

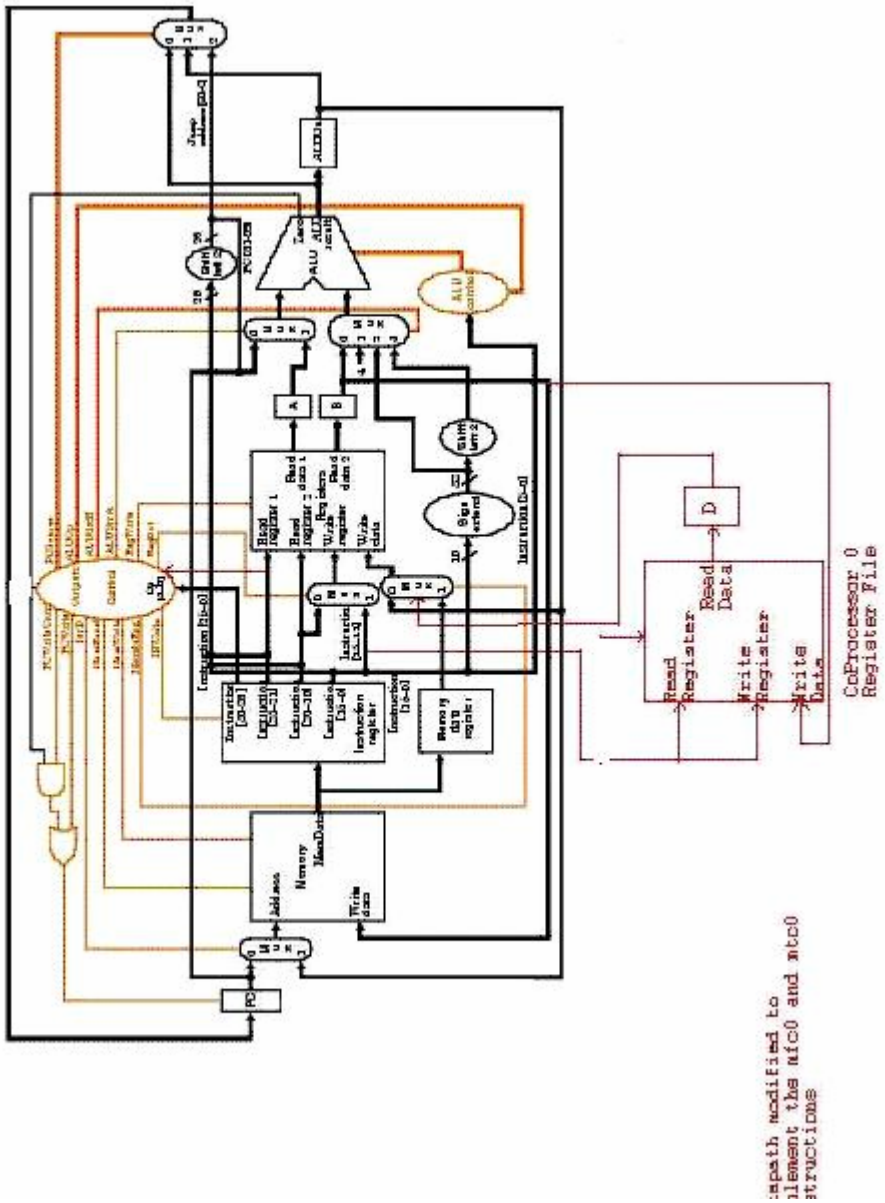
2. `mtc0 rt rd` – Move the contents of register rd to co-processor 0's register rt.

```
1. PC = PC + 4; IR = Mem[PC];
2. A = Reg[IR[25:12]];   B = Reg[IR[20:16]];
   ALUOut = PC + (SE[IR[15:0]] << 2);
   If((IR[31:26]==16)and(IR[25:21]==4))then
3. CoP0Reg[IR[15:11]] = B;
```

b. New and modified components list

Component	Inputs	Outputs	Control signals
Co-processor 0 Register file	$CP0WriteData_{31:0}$, $CP0ReadAddr_{4:0}$, $CP0WriteAddr_{4:0}$	$CP0DataOut_{31:0}$	CP0Write
D	$DIn_{31:0}$	$DOut_{31:0}$	-

c. The following datapath shows the changes required to implement the MIPS `mfc0` and `mtc0` instructions.



3. [10 points] Repeat Steps 1a and 1b for an "undefined" instruction in MIPS. Figure 5.39 in your textbook, shows the modifications required for the datapath. *Hint:* Save PC-4 in EPC, modify the PC, and update the Status Register. Read through pages 340-343 and pages A33-A35 (on the CD) of Patterson and Hennessy for more information.

   a. RTL for the undefined instruction:

```
1. PC = PC + 4; IR = Mem[PC]
2. A = Reg[IR[25:21]];   B = Reg[IR[20:16]];
   ALUOut = PC + (SE[IR[15:0]] << 2);
   If((IR[31:26]==undefined) then
3. Cause[6:2] = value to indicate an undefined
   instruction; PC = 0x8000 0180; EPC = PC – 4;
```

   b. New and modified components list

Component	Inputs	Outputs	Control signals
EPC	$EPCIn_{31:0}$,	$EPCOut_{31:0}$	EPCWrite(1 bit)
Cause	$CauseIn_{31:0}$	$CauseOut_{31:0}$	CauseWrite

   The MUX that is connected to the input of the PC is modified to be a 4-input MUX.

4. Figure 5.24 of Patterson & Hennessy, shows a complete datapath for a single-cycle implementation of most R-Type MIPS assembly language instructions, as well as `lw`, `sw`, `beq`, and `j`. For this question, you are required to modify the datapath to include the single cycle implementation of the `bne` instruction.

   a. [5 points] Write a single-cycle RTL description of an implementation of the `bne` instruction.

   b. [5 points] List all new and modified components required for the implementation of the `bne` instruction. Also, list the input, output, and control signals for each of those components. Indicate the number of bits in each signal.

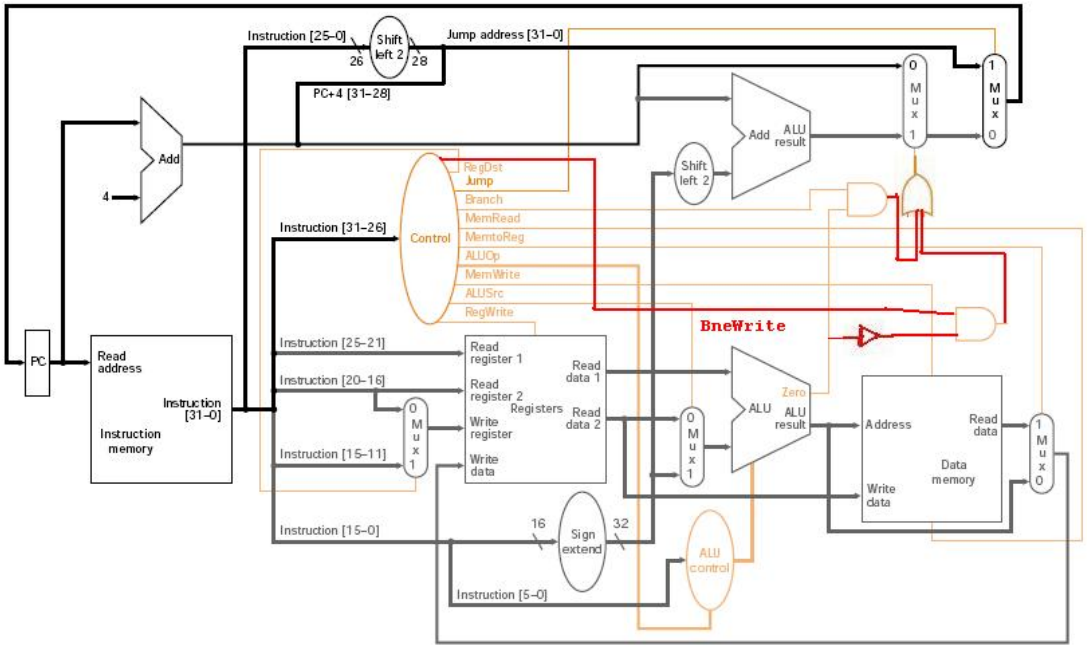
   c. [5 points] Add any necessary datapath components and control signals to the single-cycle datapath. A pdf version of Figure 5.24 is available on the class website(Homework).

   a. RTL for the `bne` instruction

```
PC = PC+4
   if ( (Mem[PC])[31:26] == 5) then
          if ( ((Mem[PC])[25:21] – (Mem[PC])[20:16]) != 0) then
                    PC = PC + SE[(Mem[PC])[15:0]] << 2
```

   b. An AND gate and a NOT gate are required.

c. The datapath modifications are shown below:



5. [15 points for Extra Credit] Consider a variant of the `lw` instruction `lwr`. The `lwr` instruction sums two registers to obtain the address of the data to be loaded and uses the R-format.

   ```
   lwr rd, rs, rt
   ```

   The instruction must read a value from memory and store it in register `rd`. The address, of the value in memory, is determined by adding the contents of registers `rs` and `rt`.

   Repeat Steps 1a, 1b and 1c for the `lwr` instruction for a multi-cycle datapath.

   a. RTL for the `lwr` instruction.

   ```
   1.  PC = PC + 4; IR = Mem[PC]
   2.  A = Reg[IR[25:21]];   B = Reg[IR[20:16]];
       ALUOut = PC + (SE[IR[15:0]] << 2);
       If((IR[31:26]==xxxx) then
   3.  ALUOut = A + B
   4.  MDR = Mem[ALUOut]
   5.  Reg[IR[15:11]] = MDR
   ```

   b. No new components required.
   c. No changes required to the datapath.